AF IFW

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Appellant:	TABBERT	Examiner:	Kiss, E.
Serial No.:	09/727,606	Group Art Unit:	2122
Filed:	December 1, 2000	Docket No.:	RA5356 (USYS.024PA)
Title:	METHOD AND APPARATUS FOR SHARING DATA STRUCTURES BETWEEN ASSEMBLY LANGUAGE PROGRAMS AND HIGH-LEVEL LANGUAGE PROGRAMS		

CERTIFICATE UNDER 37 CFR 1.8: The undersigned hereby certifies that this correspondence and the papers, as described hereinabove, are being deposited in the United States Postal Service **in triplicate**, as first class mail, in an envelope addressed to: Mail Stop Appeal Brief – Patents, Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450, on November 2, 2004.

By: Rennae Johnson
Rennae Johnson

APPEAL BRIEF

Mail Stop Appeal Brief - Patents
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Sir:

This Appeal Brief is submitted pursuant to 37 C.F.R. §41.37, in support of the Notice of Appeal filed September 2, 2004 in response to the final rejection of Claims 1-15 set forth in the final Office Action dated May 14, 2004.

Please charge Deposit Account 50-0996 (USYS.024PA) in the amount of $340.00 for filing this brief in support of an appeal. If necessary, authority is given to charge/credit Deposit Account 50-0996 (USYS.024PA) additional fees/overages in support of this filing.

I. Real Party in Interest

The real party in interest is Unisys Corporation having a place of business at 2470 Highcrest Road, Roseville, Minnesota 55113. The above referenced patent application is assigned to Unisys Corporation.

II. Related Appeals and Interferences

There are no related appeals or interferences.

III. Status of Claims

Claims 1-15 are presented for appeal. The appealed claims may be found in the attached Appendix of Appealed Claims.

IV. Status of Amendments

The Amendment and Response After Final Rejection, filed on July 12, 2004, amended the Abstract to place the application in better position for allowance or appeal. The Advisory Action dated August 13, 2004, indicated that the amendment would be entered and maintained the rejection of claims 1-15 .

V. Summary of Invention

One embodiment of Appellant's invention is directed to a method for sharing one or more high-level language data structures between an assembly language program and a high-level language program. Each data structure includes one or more elements (e.g., p. 4, ll. 6-15). The method includes determining storage requirements (e.g., FIG. 3, 252; p. 6, ll. 26-31) from high-level language definitions of the one or more data structures that are set forth in the assembly language source program (e.g., FIG. 1, 102; p. 3, l. 30 - p. 4, l. 20). The definitions are removed from the assembly source program (e.g., FIG. 1, 106; p. 5, ll. 14-20; p. 6, l. 31), and the memory addresses of the one or more data structures are determined from memory allocation directives in the assembly language source program (e.g., FIG. 3, 254; p. 7, ll. 1-5). The allocation directives are also removed from the assembly source program (e.g., FIG. 1, 106; p. 5, ll. 14-17). References to elements of the data structures in the assembly source program are replaced with memory addresses (e.g., p. 5, ll. 17-26; FIG. 3, 256; p. 7, ll. 6-11).

In another embodiment, the invention includes limitations of and related to each allocation directive including a reference to a data structure definition, a variable name, and an address (p. 4, l 25, 30; FIG. 2, 204; FIG. 3, 254).

Another embodiment includes limitations of and related to a reference to an element of a data structure in the assembly source program including one of a request for an address and a request for an offset address of the element (p. 5, l. 7-13; FIG. 2, 206; FIG. 3, 256).

Yet another embodiment is an apparatus for sharing one or more high-level language data structures between an assembly language program and a high-level language program. The apparatus includes means for determining storage requirements (e.g., FIG. 3, 252; p. 6, ll. 26-31) from high-level language definitions of the one or more data structures in an assembly language source program and removing the definitions from the assembly source program(e.g., FIG. 1, 106; p. 5, ll. 14-20; p. 6, l. 31); means for determining memory addresses of the one or more data structures from memory allocation directives in the assembly language source program (e.g., FIG. 3, 254; p. 7, ll. 1-5) and removing the allocation directives from the assembly source program (e.g., FIG. 1, 106; p. 5, ll. 14-17); and means for replacing references to elements of the data structures in the assembly source program with memory addresses (e.g., p. 5, ll. 17-26; FIG. 3, 256; p. 7, ll. 6-11).

VI. Grounds of Rejection

Claims 1-15 are mistakenly alleged to be anticipated under 35 U.S.C. §102(b) by "Turbo Assembler®Version 3.0 User's Guide," 1991 ("TASM3").

VII. Argument

The rejection of claims 1-15 under 35 U.S.C. §102(b) as being anticipated by TASM3 is improper.

The Office Actions fail to show all the limitations of the claims are taught by TASM3. A specific example from TASM3 is misapplied by the Office Actions in support of the allegation that TASM3 teaches the claim limitations. As explained below, the cited example and various teachings TASM3 clearly do not teach the claimed limitations.

i. The Office Actions fail to show that TASM3 teachings correspond to the limitations of claims 1, 2, 5, 7, 8, 9, 12, and 14

Claim 1 includes limitations of and related to determining storage requirements from high-level language definitions of the one or more data structures in an assembly language source program and removing the definitions from the assembly source program. Claim 14 is an apparatus claim with function limitations similar to the limitations of claim 1. The Office Actions do not show that TASM3 teaches high-level language definitions of the one or more data structures in an assembly language source program from which storage requirements are determined. No showing has been made of a high-level language definition of a data structure in an assembly language source program.

The Examiner misconstrues TASM3's example, resulting in a failure to show that TASM3 teaches the claim limitations. The following explanation was offered in the recent Advisory Action:

> The Examiner maintains that TASM3 discloses high-level language definitions of a data structure in the assembly language source program. In the example referenced ("EXTRN C Flag:word"), "C" is used to indicate the high-level C programming language as the format of the externally referenced code. The EXTRN directive is used within the assembly language source program to specify a definition (a data label and a corresponding reference or value for that data label) of the C language data structure "Flag".

The referenced example ("EXTRN C Flag:word") is found near the bottom of page 253 of TASM3. The teachings of TASM3 make clear that the example is an Assembler language definition of *Flag*, not a high-level language definition.

Based on TASM3's specific teachings, the variable *Flag* is defined as a *word* using an Assembler language definition. The example declares the symbol *Flag* as being referenced by a *C* module that is external to the assembly language module and is of the Assembler data type *word*. *EXTRN* indicates that *Flag* is also referenced outside the Assembler language program. *EXTRN* does not provide a data structure definition of *Flag*; the Assembler data type *word* defines *Flag*. Thus, the cited example sets forth an Assembler language definition of a variable in an assembly language program, not a high-level language definition of a data structure in an assembly language program.

Specific teachings of TASM3 further demonstrate that the Office Action is mistaken in construing the claim limitations to be taught by TASM3. As to the definition of *Flag*, TASM3 clearly explains that the example sets forth an Assembly language definition, not a high-level language definition., At page 255 TASM3 shows a table of C++ data types and corresponding Assembler data types. From the table it is apparent that the cited example uses an Assembler data type, not a high-level language type. Since *Flag* is not defined as one of the high-level language data types: *enum*, *unsigned short*, *short*, *unsigned int*, *int*, or *near**, it is clear that TASM3 does not teach the high-level language definition of a data structure in the assembly language source program. The example is further problematic because a syntactically correct C-language definition of *Flag* would not have a ":" separating *Flag* from *word*.

TASM3's teachings clearly explain that the high-level language definition of a variable is external to the Assembler language code, not in the Assembler language code. For example, at page 225 TASM3 teaches, "External symbols are symbols that are defined outside a module, that you can use within a module." Thus, the cited example declares the symbol *Flag* as being defined by a *C* module that is external to the assembly language module, and the high-level language definition of *Flag* is not in the Assembler language code.

The Office Actions have not established that TASM3 has a high-level language definition of a data structure in the assembly language program from which storage requirements are determined.

The further limitations of claim 1, are also not shown by TASM3. Since TASM3 does not have high-level language data structure definitions in the assembly language source file, TASM3 does not teach removing the definitions from the assembly source program.

Claim 1 includes further limitations of and related to separate memory allocation directives for allocating memory for the data structures. The Office Actions have not shown any teachings of TASM3 to correspond to these limitations. The claims make clear that an allocation directive is not one and the same with the high-level language definition of a data structure. However, the Office Actions do not explain, and it is not apparent, which teachings of TASM3 are used to support the alleged anticipation of allocation directives as compared to the teachings used to support the alleged anticipation of the high-level language definitions of the data structure. Thus, the limitations of and related to the separate memory allocation directives for allocating memory for the data structures are not shown to be taught by TASM3.

Claims 2, 5, 7, 8, 9, and 12 depend directly or indirectly from claim 1 and are not shown to be anticipated by TASM3 for at least the reasons set forth above.

ii. The Office Actions fail to show TASM3 teachings corresponding to the limitations of claims 3, 4, 10, 11, 13, and 15

Claims 3 and 10 include limitations of and related to each allocation directive including a reference to a data structure definition, a variable name, and an address. The Office Actions again misapply TASM3's teachings in alleging that claim 3 is anticipated. Specifically, the Office Actions rely on the TASM's EXTRN as being a directive to reference an external C++ variable. However, none of the teachings of TASM3 have been cited to show that use of EXTRN includes an address along with the variable name. Furthermore, since EXTRN is used to reference an external variable, there would be no apparent need for an EXTRN statement to include an address along with a variable name. Therefore, the Office Actions fail to establish that TASM3 anticipated claim 3.

Claims 4, 11, 13, and 15 depend from directly or indirectly from claims 3 or 10 and are not shown to be anticipated by TASM3 for at least the reasons set forth above.

iii. The Office Actions fail to show that TASM3 teachings correspond to the limitations of claim 6

Claim 6 includes limitations of and related to a reference to an element of a data structure in the assembly source program including one of a request for an address and a request for an offset address of the element. The Office Actions cite a SIZE function provided by TASM3 as illustrated by TASM3's Table 5.12 on page 91. However, the explanations in the table clearly indicate that byte counts are returned by the function, not an address or an offset. Therefore, the Office Actions fail to show that claim 6 is anticipated by TASM3.

VIII. Conclusion

In view of the above, Appellant respectfully submits that the claimed invention is patentable over the cited prior art, and that the rejections of Claims 1-15 are in error. Appellant respectfully requests reversal of the rejections as applied to the appealed claims and allowance of the application.

Respectfully submitted,

CRAWFORD MAUNU PLLC
1270 Northland Drive – Suite 390
St. Paul, MN 55120
(651) 686-6633

By: _____
Name: LeRoy D. Maunu
Reg. No.: 35,274

APPENDIX OF APPEALED CLAIMS (09/727,606)

1. A computer-implemented method for sharing one or more high-level language data structures between an assembly language program and a high-level language program, each data structure including one or more elements, comprising the steps of:

determining storage requirements from high-level language definitions of the one or more data structures in an assembly language source program and removing the definitions from the assembly source program;

determining memory addresses of the one or more data structures from memory allocation directives in the assembly language source program and removing the allocation directives from the assembly source program; and

replacing references to elements of the data structures in the assembly source program with memory addresses.

2. The method of claim 1, further comprising:

assembling the assembly source program into a first object code segment;

compiling the high-level language program into a second object code segment; and

linking the first and second object code segments.

3. The method of claim 1, wherein each allocation directive includes a reference to a data structure definition, a variable name, and an address.

4. The method of claim 3, wherein variable name includes an array size.

5. The method of claim 1, wherein a reference to an element of a data structure in the assembly source program includes a hierarchical specification of the element.

6. The method of claim 5, wherein a reference to an element of a data structure in the assembly source program includes one of a request for an address and a request for an offset address of the element.

7. The method of claim 5, wherein a reference to an element of a data structure in the assembly source program includes one of a request for an address of the element, a request for an offset address of the element, and a request for a size of the element.

8. The method of claim 7, further comprising replacing a reference to an element of a data structure that includes a request for a size of an element with the size of the element.

9. The method of claim 8, further comprising:
assembling the assembly source program into a first object code segment;
compiling the high-level language program into a second object code segment; and
linking the first and second object code segments.

10. The method of claim 8, wherein each allocation directive includes a reference to a data structure definition, a variable name, and an address.

11. The method of claim 10, wherein variable name includes an array size.

12. The method of claim 5, further comprising:
assembling the assembly source program into a first object code segment;
compiling the high-level language program into a second object code segment; and
linking the first and second object code segments.

13. The method of claim 5, wherein each allocation directive includes a reference to a data structure definition, a variable name, and an address.

14. An apparatus for sharing one or more high-level language data structures between an assembly language program and a high-level language program, each data structure including one or more elements, comprising:

means for determining storage requirements from high-level language definitions of the one or more data structures in an assembly language source program and removing the definitions from the assembly source program;

means for determining memory addresses of the one or more data structures from memory allocation directives in the assembly language source program and removing the allocation directives from the assembly source program; and

means for replacing references to elements of the data structures in the assembly source program with memory addresses.

15. The method of claim 13, wherein variable name includes an array size.